Package 'whoa'

August 11, 2021

Type Package

Title Evaluation of Genotyping Error in Genotype-by-Sequencing Data

Version 0.0.2

Maintainer Eric C. Anderson <eric.anderson@noaa.gov>

Description This is a small, lightweight package that lets users investigate the distribution of genotypes in genotype-by-sequencing (GBS) data where they expect (by and large) Hardy-Weinberg equilibrium, in order to assess rates of genotyping errors and the dependence of those rates on read depth. It implements a Markov chain Monte Carlo (MCMC) sampler using 'Rcpp' to compute a Bayesian estimate of what we call the heterozygote miscall rate for restriction-associated digest (RAD) sequencing data and other types of reduced representation GBS data. It also provides functions to generate plots of expected and observed genotype frequencies. Some background on these topics can be found in a recent paper ``Recent advances in conservation and population genomics data analysis'' by Hendricks et al. (2018) <doi:10.1111/eva.12659>, and another paper describing the MCMC approach is in preparation with Gordon Luikart and Thierry Gosselin.

License CC0

Encoding UTF-8

LazyData TRUE

Depends R (>= 3.3.0)

Imports dplyr, magrittr, tibble, tidyr, Rcpp (>= 0.12.16), vcfR, viridis, ggplot2

LinkingTo Rcpp

RoxygenNote 7.1.1

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation yes

Author Eric C. Anderson [aut, cre]

Repository CRAN

Date/Publication 2021-08-11 06:10:02 UTC

R topics documented:

`exp_and_obs_geno_freqs` *Computed expected and observed genotype frequencies from a 'vcfR' object*

Description

Under the assumption of Hardy-Weinberg equilibrium, this function uses the estimated allele frequencies from the data set in v to compute expected genotype frequencies, and then reports these along with the observed genotype frequencies. Loci come out named as CHROM–POS.

Usage

```
exp_and_obs_geno_freqs(
  v = NULL,
  d012 = NULL,
  prop_indv_required = 0.5,
  prop_loci_required = 0.5
)
```

Arguments

`v`	a 'vcfR' object. Exactly one of `v` or `d012` is required. But you can't use both!
`d012`	an integer matrix (or a numeric matrix, which will be coerced to be of integer type) with individuals in columns, and markers in rows. 0 denotes a genotype homozygous for the reference allele, 1 is a heterozygote, 2 is a homozygote for the alternate allele, and -1 denotes missing data. This matrix is not required to have column (sample) names. They won't be used if they are present. But, the matrix must have rownames, which should be in the format of CHROM–POS (i.e. the "chromosome" name (or the "contig" name) followed by a "–" followed by the position of the marker in the "chromosome"). Exactly one of `v` or `d012` is required. But you can't use both!
`prop_indv_required`	loci will be dropped if a proportion of individuals less than prop_indv_required have non-missing data at that locus. Default is 0.5

prop_loci_required
: individual will be dropped if their proportion of non-missing loci is less than prop_loci_required. Default is 0.5.

Value

Returns a tibble with the following columns: snp = the locus name as CHROM–POS; p = The frequency of the alternate (ALT) allele; ntot = the total number of individuals with no missing data at the locus; geno = column telling which genotype (0, 1, or 2) is referred to; p_exp = expected frequency of the genotype; p_obs = observed frequency of genotype; n_exp = expected number of such genotypes; n_obs = observed number of such genotypes; z_score = simple statistic giving how far the observed genotype frequency is from that expected under Hardy-Weinberg equilibrium.

Examples

```
eao <- exp_and_obs_geno_freqs(v = lobster_buz_2000)

# if you wanted to run that on an 012 matrix,
# it would be like this:
eao012 <- exp_and_obs_geno_freqs(d012 = lobster_buz_2000_as_012_matrix)
```

---

geno_freqs_scatter *return a 'ggplot2' plot object of observed and expected genotype freqs*

Description

return a 'ggplot2' plot object of observed and expected genotype freqs

Usage

```
geno_freqs_scatter(gfc, alpha = 0.2, max_plot_loci = 500)
```

Arguments

gfc	a tibble like that created by exp_and_obs_geno_freqs()
alpha	the transparency (alpha) parameter to apply to the points in the scatterplot. Default is 0.2.
max_plot_loci	By default this plots only 500 loci, sampled randomly, to keep 'ggplot2' taking forever to plot, for example, 100K points. If you want to plot all the points, set this to a number larger than the number of single nucleotide polymorphisms (SNPs) in the data set.

Examples

```
# get the expected and observed geno freqs
gfreqs <- exp_and_obs_geno_freqs(lobster_buz_2000)
g <- geno_freqs_scatter(gfreqs)

# now g is a 'ggplot2' object.
```

infer_m *get posterior estimates for m from different read depth categories*

Description

This function calls internal C++ routines that perform Markov chain Monte Carlo (MCMC) to sample from the posterior distribution of the heterozygote miscall rate for each read depth category.

Usage

```
infer_m(
  v,
  minBin,
  indivs = NULL,
  init_m = 0.1,
  num_sweeps = 500,
  burn_in = 100
)
```

Arguments

v	a 'vcfR' object holding the information from a variant call format (VCF) file with the genotype and depth data. If you are going to be breaking the estimates down by read depth categories, the VCF file must have a DP field for every genotype.
minBin	minimum number of observations for each read depth bin. If you have 10K markers and 50 individuals then you have about 500,000 genotypes to play with. Requiring bins with at least 5,000 genotypes in them will give you less than 100 bins. You can play around with this number to get the right number of bins. The algorithm breaks the read depths up into bins that have at least minBin genotypes in them.
indivs	a character vector holding the names of the individuals from the VCF file to include in the analysis. By default this is NULL, in which case everyone from the file is included.
init_m	the initial value of the heterozygote miscall rate assumed for each read depth bin. By default this is 0.1.
num_sweeps	the number of sweeps of the MCMC algorithm to do. Default is 500, which is a little on the short side. Run multiple times and make sure the values obtained are similar across runs to assess convergence.
burn_in	how many sweeps from the beginning of the chain to discard when computing the posterior means and quantiles. Default is 100. Note that full traces of the visited m values for every read depth bin are returned as well, so that the behavior of the chain in those early steps can be investigated.

Details

The read depth bins are determined by passing to the function `minBin`—the minimum number of observations desired for each read depth bin. The function then breaks the observations into bins so that each read depth bin has at least minBin observations.

Note that if you want to estimate the heterozygote miscall rate **overall** (i.e., not conditioning each estimate on a read depth bin), then simply give a very large number (larger than the number of markers times the number of individuals) for minBin. For example, you could use a number like 1e15 for minBin. As a consequence, all the genotypes will be put into a single read depth bin.

Value

A list with six components:

m_posteriors A tibble with 6 columns: bin = the index of the read depth bin; mean = the posterior mean estimate of the the heterozygote miscall rate in that bin; lo95 = the low endpoint of the 95 95 and mean_dp = the mean read depth in the bin.

m_traces A tibble with all the values visited for m for every read depth bin. This tibble has three columns: bin = the index of the read depth bin; sweep = the sweep number, value = the value of m for that read depth bin in that particular sweep.

dp_summary A tibble summarizing how many genotypes of different read depths appear in each bin.

bin_stats A tibble with a different summary of the read-depth bins.

num_sweeps Number of MCMC sweeps used.

burn_in Number of sweeps discarded as burn in.

Examples

```
# Shorter run than recommended (for quick example...)
im <- infer_m(lobster_buz_2000, minBin = 1000, num_sweeps = 100, burn_in = 20)
```

---

lobster_buz_2000 *Restriction-associated digest (RAD) sequence data from 36 lobsters at 2000 single nucleotide polymorphisms (SNPs)*

Description

A data set from the study by Benestan et al. (2016).

Usage

```
lobster_buz_2000
```

Format

A 'vcfR' object in which the original data set has been reduced to just the 36 lobsters from the BUZ population and a randomly sampled 2000 SNPs from the 10,156 originally available.

This is the sort of object obtained after calling `vcfR::read.vcfR()` on a variant call format (VCF) file.

Source

https://datadryad.org/stash/dataset/doi:10.5061/dryad.q771r

---

lobster_buz_2000_as_012_matrix

An 012 matrix from (RAD) sequence data from 36 lobsters at 2000 single nucleotide polymorphisms (SNPs)

Description

A data set from the study by Benestan et al. (2016).

Usage

```
lobster_buz_2000_as_012_matrix
```

Format

This is an integer matrix with positions in rows and samples in columns. It is an 012 matrix that corresponds to lobster_buz_2000. It is useful as an example of the necessary format for the d012 argument to `exp_and_obs_geno_freqs`.

Source

https://datadryad.org/stash/dataset/doi:10.5061/dryad.q771r

---

make_it_012 *just a quick function for making an 012 matrix from a character matrix*

Description

The standard way within R of pulling values out of a named vector really bogs down on large data sets. So I will do this instead.

Usage

```
make_it_012(M)
```

Arguments

M	a character matrix of variant call format (VCF) genotypes and no dimnames. Allowable values are "0/0", and "0|0", which get coverted to integer 0; "0/1", "0|1", "1/0", and "1|0", which get converted to integer 1; and "1/1", and "1|1", which get converted to integer 2. Everything else gets converted to -1 to denote missing data.

Value

An integer matrix of values which are 0, 1, 2, or -1.

Examples

```
# get an 012 matrix from the lobster data
tmp <- t(vcfR::extract.gt(lobster_buz_2000, element = "GT"))
dimnames(tmp) <- NULL
g <- make_it_012(tmp)
```

---

posteriors_plot *return a 'ggplot2' plot object of the posterior esimates for heterozyote miscall rates*

Description

This just returns a 'ggplot2' plot object that plots the read depth bins on the x-axis and the posterior mean m estimates (and credible intervals) on the y-axis, and depicts the number of genotypes in each read depth bin using color.

Usage

```
posteriors_plot(P)
```

Arguments

P	the tibble that is the m_posteriors component of infer_m

Value

a 'ggplot2' plot object.

Examples

```
# get something to plot (short run for example)
im <- infer_m(lobster_buz_2000, minBin = 1000, num_sweeps = 100, burn_in = 20)

# then plot it
g <- posteriors_plot(im$m_posteriors)

# now g is a 'ggplot2' object
```

whoa *whoa: Evaluation of genotyping error in genotype by sequencing data*

Description

The name is found in the capitals here: Where's my Heterozygotes?! Observations of genotyping Accuracy.

the whoa main high-level functions

The main function in the package whoa is infer_m. This function infers the heterozygote miscall rate (the rate at which true heterozygotes have been miscalled as homozygotes in genotype-by-sequencing data) for calls made upon genotypes falling within different read depth categories.

The output from infer_m is easily plotted by passing the m_posteriors component of the output list from infer_m into posteriors_plot.

example data

The package comes with a small data set, lobster_buz_2000, which was read in from a variant call format (VCF) file and is now stored in the package as a 'vcfR' object.

Index